Reference [3]

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) **Offenlegungsschrift**

(10) **DE 196 35 429 A 1**

(51) Int. Cl.[6]: G 06 F 17/30

(21) Aktenzeichen: 196 35 429.3
(22) Anmeldetag: 2. 9. 96
(43) Offenlegungstag: 5. 3. 98

DE 196 35 429 A 1

≙ US 6,510,435

(71) Anmelder:
Bayer, Rudolf, Prof., 82194 Gröbenzell, DE

(74) Vertreter:
Patentanwälte Raeck & Hössle, 70182 Stuttgart

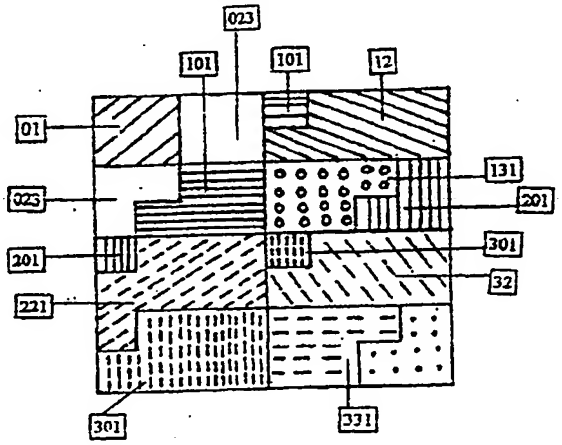
(72) Erfinder:
gleich Anmelder

(56) Entgegenhaltungen:
DE 39 08 684 A1

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Datenbanksystem und Verfahren zum Verwalten eines n-dimensionalen Datenbestands

(57) Datenbanksystem und Verfahren zum Verwalten eines multidimensionalen Datenbestands. Das Datenbanksystem umfaßt eine Recheneinrichtung, einen Hauptspeicher und eine insbesondere periphere Speichereinrichtung, wobei zum Indizieren und Speichern des in einem mehrdimensionalen Würfel liegenden Datenbestands auf Speicherseiten gegebener Speicherkapazität des peripheren Speichermittels ein wiederholtes iteratives Unterteilen des mehrdimensionalen Würfels in allen Dimensionen in Subwürfel erfolgt, bis aufeinanderfolgende Subwürfel zu Regionen zusammenfaßbar sind, die jeweils eine Menge von Datenobjekten beinhalten, die auf einer der Speicherseiten gegebener Speicherkapazität abspeicherbar sind. Mit den erfindungsgemäßen Verfahren zum Verwalten, Einfügen, Löschen und Suchen von Datenobjekten wird eine als FB-Baum bezeichnete dynamische Datenstruktur bereitgestellt, mit deren Hilfe verbesserte Zugriffzeiten erzielt werden und die somit zur Verwendung in Online-Anwendungen geeignet ist.



DE 196 35 429 A 1

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

BUNDESDRUCKEREI 01. 98 702 070/437

Beschreibung

Die vorliegende Erfindung betrifft ein Datenbanksystem sowie ein Verfahren zum Verwalten eines in einem n-dimensionalen Würfel mit $n > 1$ liegenden Datenbestands.

Zur Verwaltung umfangreicher eindimensionaler Datenmengen auf Massenspeichern, wie beispielsweise Magnetplattenspeichern ist als Datenstruktur der sogenannte B-Baum (oder auch B*-Baum oder Präfix-B-Banm) bekannt. Die Datenstruktur des B-Baums hat gegenüber derjenigen eines einfachen Suchbaums den Vorteil, daß beim Datenzugriff geringere Suchzeiten notwendig sind. Die Suchzeit, die sich zum Auffinden gewisser Daten ergibt, beträgt bei einem einfachen Suchbaum mit n Knoten mindestens $\log_2(n)$ Schritte. Bei einem Suchbaum mit 1.000.000 Knoten muß man daher mit $\log_2(1.000.000) \approx 20$ Plattenzugriffen rechnen. Nimmt man eine mittlere Zugriffszahl von 0,1 Sek. an, so benötigt die Suche eines Knotens 2 Sek. Dieser Wert ist für die Praxis zu groß. Bei der Datenstruktur des B-Baums wird die Zahl der Plattenzugriffe reduziert, indem man je Zugriff nicht einen einzelnen Knoten, sondern ein ganzes, einem Knoten zugeordnetes Teilstück von der Magnetplatte in den Hauptspeicher überträgt und innerhalb dieses Teilstücks sucht. Teilt man beispielsweise den B-Baum in Bereiche zu je sieben Knoten und überträgt mit jedem Plattenzugriff einen solchen Bereich in den Hauptspeicher, so reduziert sich die Zahl der Plattenzugriffe für die Suche eines Knotens von maximal 6 auf maximal 2. Bei 1.000.000 Knoten benötigt man somit nur noch $\log_8(1.000.000) = 7$ Zugriffe. In der Praxis unterteilt man den Suchbaum meistens in Teilbereiche der Größe 2^8-1 bis $2^{10}-1$ Knoten. Bei einer Bereichsgröße von 255 Knoten benötigt man für die Suche eines Knotens in einem Baum mit 1.000.000 Knoten $\log_8$(1 Mio) $\approx$ 2,5 Plattenzugriffe, so daß die Suche nach einem gegebenen Wert nur noch etwa 0,3 Sek. dauert. Die Suchzeit innerhalb eines Teilbereichs mit 255 Knoten, der sich im Hauptspeicher befindet, kann gegenüber dem Plattenzugriff vernachlässigt werden. Der B-Baum ist ein höhen-balancierter Baum, bei dem alle Blätter auf dem gleichen Niveau liegen.

Zur Verwaltung eines multi-dimensionalen Datenbestands sind aus K. Mehlhorn: Multidimensional Searching und Computational Geometry, Springer, Heidelberg 1984, die sogenannten dd-Bäume bekannt. Mit den dd-Bäumen lassen sich prinzipiell drei Arten von Anfragen durchführen, nämlich Punkt-Anfragen, Bereichs-Anfragen und Anfragen, bei denen manche Intervalle als (−unendlich, +unendlich) angegeben sind. Die Datenstruktur eines dd-Baums gestattet jedoch nur bei Punktanfragen einen schnellen Zugriff da dann nur ein Pfad im Baum abgesucht werden muß. Bei den anderen Anfragen kann es geschehen, daß der gesamte Baum durchsucht werden muß. Darüber hinaus sind dd-Bäume statisch, d. h. die gesamte zu verwaltende Objektmenge muß schon bekannt sein, bevor der dd-Baum aufgebaut werden kann. Bei den meisten Anwendungen in der Praxis ist aber die Objektinenge dynamisch, d. h. Objekte müssen in beliebiger Reihenfolge und jederzeit im Baum eingefügt und gelöscht werden können, ohne daß der ganze Baum neu aufgebaut werden muß. Des weiteren eignen sich dd-Bäume nur für Hauptspeicheranwendungen, jedoch nicht für Peripheriespeicher, die zum Abspeichern sehr großer Datenmengen benötigt werden.

In "The Grid File" von Nievergelt et al, ACM TODS, Bd. 9, Heft 1, März 1984 sind zur Verwaltung multidimensionaler Daten sogenannte Grid-Files beschrieben, bei denen Anfragen für Punkte und Bereiche auf der Basis einer Inhaltsverzeichnis-Struktur, dem sogenannten Grid, durchgeführt werden. Obwohl diese Datenverwaltung eine schnelle Suche bei Punkt- und Bereichsanfragen gewährleistet, handelt es sich um ein statisches Verfahren, so daß beim dynamischen Einfügen und Löschen von Datenobjekten regelmäßig die gesamte Inhaltsverzeichnis-Struktur völlig umorganisiert werden muß. Somit ist dieses Verfahren für viele Anwendungen, insbesondere für Online-Anwendungen nicht geeignet.

Aus A. Guttmann: A dynamic Index Structure for spatial Searching, Proceedings ACM SIGMOD Intl. Conference on Management of Data, 1984, Seiten 47—57, sind als Datenstruktur zur Verwaltung multidimensionaler Daten sogenannte R-Bäume bekannt. Diese Bäume, die hauptsächlich bei sogenannten Geo-Datenbanken Anwendung finden, sind ähnlich wie B-Bäume in der Höhe balanciert und gestatten auch dynamisches Einfügen und Löschen von Objekten. Bei der Beantwortung von Anfragen sind jedoch keine schnellen Zugriffszeiten garantiert, weil unter Umständen beliebig viele Pfade in dem betreffenden Baum, im Extremfall sogar der gesamte Baum, durchsucht werden müssen, um eine Anfrage zu beantworten. Somit sind diese R-Bäume für die meisten Online-Anwendungen nicht geeignet.

Das heute in der Praxis am weitesten verbreitete Verfahren zur Verwaltung eines multidimensionalen Datenbestands basiert auf den ursprünglichen eindimensionalen B-Bäumen, wobei für jede Dimension des Ausgangsdatenbestands jeweils ein B-Baum eingesetzt wird so daß Bereichsanfragen in einem n-dimensionalen Datenbestand durch n B-Bäume unterstützt werden. Bei einer Bereichsanfrage werden somit für jede Dimension sämtliche Objekte vom peripheren Speicher geholt, deren Werte innerhalb des in der Anfrage spezifizierten Intervalls für diese Dimension liegen. Diese Datenobjekte bilden die Treffermenge in der betreffenden Dimension. Um die gewünschte Antwortmenge zu bestimmen, muß eine Schnittmenge der Treffermengen sämtlicher Dimensionen berechnet werden, was üblicherweise erst eine Sortierung dieser Mengen erfordert. Auch beim Einfügen und Löschen eines Datenobjekts müssen entsprechend n B-Bäume durchsucht und modifiziert werden.

Ausgehend hiervon liegt der Erfindung die Aufgabe zugrunde, ein Datenbanksystem und ein Verfahren zum Verwalten eines n-dimensionalen Datenbestands bereitzustellen, das aufgrund verbesserter Zugriffszeiten insbesondere zur Verwendung in Online-Anwendungen geeignet ist und das ein dynamisches Einfügen und Löschen von Datenobjekten gestattet.

Zur Lösung der Aufgabe wird erfindungsgemäß ein Datenbanksystem mit den Merkmalen des Anspruches 1 vorgeschlagen. Das erfindungsgemäße Datenbanksystem umfaßt eine Recheneinrichtung, einen Hauptspeicher und eine Speichereinrichtung, bei der es sich insbesondere um eine periphere Speichereinrichtung handelt. Der Grundgedanke der Erfindung liegt darin, einen zu verwaltenden multidimensionalen Datenbestand in einen mehrdimensionalen Würfel zu legen und zum Indizieren und Speichern dieses Datenbestands mittels der

Recheneinrichtung ein wiederholtes iteratives Unterteilen des mehrdimensionalen Würfels in allen Dimensionen in Subwürfel vorzunehmen. Das Unterteilen wird dabei so oft wiederholt, bis aufeinanderfolgende Subwürfel zu Regionen zusammenfaßbar sind die jeweils eine Menge von Datenobjekten beinhalten, die auf einer der Speicherseiten gegebener Speicherkapazität der insbesondere peripheren Speichereinrichtung abspeicherbar sind. Da die Regionen aus aufeinanderfolgenden Subwürfeln zusammengefaßt sind, liegen auch die Regionen hintereinander, so daß sie eine eindimensionale Struktur bilden. Somit ist erfindungsgemäß bei einem Einfügen bzw. Löschen von Datenobjekten lediglich die Modifikation einer einzigen Datenstruktur, beispielsweise eines Baumes, erforderlich.

In Ausgestaltung der Erfindung erfolgt das Abspeichern der Datenobjekte einer Region auf einer Speicherseite gegebener Speicherkapazität unter Zuordnung eines Zeigers zu der Speicherseite und einer die Regionengrenzen bezeichnenden Adresse. Somit sind jeder abzuspeichernden Region eindeutige, die Regionengrenzen definierende Adressen sowie ein auf die Speicherseite, auf welcher die betreffende Region abgespeichert ist, verweisender Zeiger zugeordnet. Dadurch wird das Auffinden der Region und der in der Region enthaltenen Datenobjekte bei Verwaltungsvorgängen wie Beantwortung von Anfragen, Löschen oder Einfügen von Datenobjekten erleichtert.

In weiterer Ausgestaltung der Erfindung erfolgt die Speicherung des Zeigers und der Adresse in einem B-Baum, B*-Baum oder Präfix-B-Baum, so daß bei einer Adreßsuche eine einfache und schnell durchzuführende Suche in einem B-Baum zur Identifikation der benötigten Region über den der Adresse zugeordneten und auf die Speicherseite der benötigten Region verweisenden Zeiger erfolgen kann.

In weiterer Ausgestaltung der Erfindung erfolgt die Speicherung der Datenobjekte selbst in den Blattseiten des B-Baums, B*-Baums oder Präfix-B-Baums.

In vorteilhafter Ausgestaltung der Erfindung besteht die die Regionengrenzen bezeichnende Adresse aus Angaben über den letzten der die Region bildenden Subwürfein. Besonders vorteilhaft erweist sich ein Datenbanksystem, bei dem die Adresse Angaben über die Anzahl der auf jeder Unterteilungsstufe in der Region enthaltenen Subwürfel umfaßt. Eine Region ist somit eindeutig bezeichnet, wenn der letzte vollständig in der Region enthaltene Subwürfel durch die Adreß-Angabe ebenfalls eindeutig bezeichnet ist. Der Beginn der Region wird dabei durch die Adreß-Angabe über den letzten der die vorhergehende Region bildenden Subwürfeln angegeben.

Zur weiteren Lösung der der Erfindung zugrundeliegenden Aufgabe wird ein Verfahren zum Verwalten eines in einem n-dimensionalen Würfel mit $n>1$ liegenden Datensbestands mit den Merkmalen des Anspruchs 8 vorgeschlagen. Mit dem erfindungsgemäßen Verfahren wird zum Indizieren und Speichern eines multidimensionalen Datenbestands dieser Datenbestand in einen n-dimensionalen Würfel mit $n>1$ gelegt. Dieser Würfel bildet in seiner Gesamtheit eine Ausgangsregion, die sämtliche Datenobjekte des Datenbestands enthält. Sollte die Anzahl der vorhandenen Datenobjekte kleiner oder gleich sein als die der vorgegebenen Speicherkapazität einer Speicherseite entsprechende Zahl von Datenobjekten, so wird die Ausgangsregion auf einer Speicherseite abgespeichert. Andernfalls wird die Ausgangsregion entlang einer Spaltadresse gespalten, wobei die Spaltadresse so gewählt wird, daß etwa entlang der Datenmitte zwei neue Teilregionen erzeugt werden. Mit jeder dieser Teilregionen wird anschließend so verfahren, wie zuvor mit der Ausgangsregion, d. h. es wird die Anzahl der jeweils in der Teilregion enthaltenen Datenobjekte bestimmt und mit der der vorgegebenen Speicherkapazität einer Speicherseite entsprechenden Zahl verglichen. Ist der Datenbestand nicht größer als die der vorgegebenen Speicherkapazität entsprechende Zahl, so wird die betreffende Region auf einer Speicherseite abgespeichert, andernfalls wird sie wiederum entlang der Datenmitte gespalten und das Verfahren beginnt von neuem.

Vorteilhafterweise erfolgt das Abspeichern der Datenobjekte einer Region oder Teilregion unter paralleler Abspeicherung einer der betreffenden Region zugeordneten Adresse sowie eines der Adresse zugeordneten und auf die Speicherseite, auf der die abgespeicherten Datenobjekte enthalten sind, verweisenden Zeigers. Bei der parallel abzuspeichernden Adresse kann es sich vorteilhafterweise um die Spaltadresse handeln, die das Ende der einen und den Anfang der anderen Region angibt.

In Ausgestaltung der Erfindung erfolgt die Speicherung der Adresse und des Zeigers in einem B-Baum, B*-Baum oder Präfix-B-Baum, wobei durch aufeinanderfolgende Adressen jeweils Regionen definiert werden, deren Datenobjekte jeweils auf einer Speicherseite gegebener Speicherkapazität abgespeichert sind.

Zur weiteren Lösung der der Erfindung zugrundeliegenden Aufgabe wird des weiteren ein Verfahren zum Einfügen von Datenobjekten in einen in Regionen aufgeteilten und auf Speicherseiten gegebener Speicherkapazität einer insbesondere peripheren Speichereinrichtung abgespeicherten n-dimensionalen Datenbestand mit $n>1$ mit den Merkmalen des Anspruches 11 vorgeschlagen. Vorteilhafterweise handelt es sich bei dem abgespeicherten n-dimensionalen Datenbestand um einen gemäß dem vorstehend beschriebenen erfindungsgemäßen Verfahren indizierten und abgespeicherten Datenbestand. Erfindungsgemäß wird ausgehend von den Koordinaten des einzufügenden Datenobjekts eine das Datenobjekt enthaltende Region des n-dimensionalen Datenbestands sowie die Speicherseite, auf der diese Region abgespeichert ist, ermittelt. Daraufhin werden die auf dieser Speicherseite gespeicherten Datenobjekte gezählt. Sollte die Anzahl der gespeicherten Datenobjekte kleiner sein als die der gegebenen Speicherkapazität der Speicherseite entsprechende Anzahl, so wird das einzufügende Datenobjekt ebenfalls auf dieser Speicherseite abgespeichert. Anderenfalls wird eine Spaltadresse für die auf dieser Speicherseite gespeicherte Region derart ausgewählt, daß durch Spalten der Region entlang dieser Spaltadresse eine erste und eine zweite Teilregion erzeugt werden, in welchen jeweils weniger als etwa die Hälfte der der gegebenen Speicherkapazität entsprechenden Anzahl an Datenobjekten enthalten ist. Dann wird das einzufügende Datenobjekt in diejenige Teilregion eingefügt, in welcher die Koordinaten des Datenobjekts liegen, woraufhin die erste und die zweite Teilregion auf jeweils einer Speicherseite abgespeichert werden.

Erfindungsgemäß ist somit in der gegebenen Datenstruktur ein dynamisches Einfügen von Datenobjekten möglich, ohne daß die Gesamtheit der Datenstruktur modifiziert oder neu angelegt werden muß. Sollte durch

das Einfügen des neuen Datenobjekts die Region, in welche die Einfügung vorgenommen wurde, nicht mehr auf einer Speicherseite abspeicherbar sein, so wird diese Region in zwei weitere Regionen aufgespalten, wodurch lediglich die betreffende zu spaltende Region bzw. die durch die Spaltung neu entstehenden Teilregionen modifiziert und neu abgespeichert werden müssen.

Vorteilhafterweise erfolgt das Auffinden der Speicherseite in dem erfindungsgemäßen Verfahren zum Einfügen von Datenobjekten mittels in einem B-Baum, B*-Baum oder Präfix-B-Baum gespeicherter, den Speicherseiten zugeordneter Adressen und Zeiger. Dadurch gestaltet sich das Auffinden der gesuchten Speicherseite besonders einfach und schnell. Entsprechend erweist es sich als vorteilhaft, wenn das Speichern der neu entstandenen Teilregionen unter Ersetzung des vormaligen Zeigers und der Adresse der gespalteten Region durch jeweils der ersten und der zweiten Teilregion Zugeordnete Adressen und Zeiger erfolgt. Dabei kann beispielsweise für die erste Teilregion als begrenzende Adresse die Spaltadresse verwendet werden und für die zweite Teilregion kann die begrenzende Adresse der gespaltenen Region verwendet werden.

Als besonders vorteilhaft erweist es sich, wenn die Speicherung der Adresse und des Zeigers in einem B-Baum, B*-Baum oder Präfix-B-Baum erfolgt, wobei durch aufeinanderfolgende Adressen jeweils Regionen definiert werden, deren Datenobjekte jeweils auf einer Speicherseite gegebener Speicherkapazität abgespeichert sind.

Zur weiteren Lösung der der Erfindung zugrundeliegenden Aufgabe wird ein Verfahren zum Löschen von Datenobjekten mit den Merkmalen des Anspruches 15 vorgeschlagen. Demnach werden ausgehend von den Koordinaten des zu löschenden Datenobjektes die das Datenobjekt enthaltende Region des n-dimensionalen Datenbestands sowie die Speicherseite, auf welcher diese Region abgespeichert ist, bestimmt und das zu löschende Objekt wird auf dieser Speicherseite gelöscht. Danach wird die Anzahl der auf dieser Speicherseite gespeicherten Datenobjekte bestimmt und die Region wird mit einer ihrer beiden Nachbarregionen verschmolzen, falls die Anzahl der gespeicherten Datenobjekte kleiner ist als etwa die Hälfte der der gegebenen Speicherkapazität der Speicherseite entsprechenden Anzahl. Daraufhin wird wiederum die Anzahl der in der durch das Verschmelzen neu entstandenen Region vorhandenen Datenobjekte bestimmt. Ist diese Anzahl kleiner als die der gegebenen Speicherkapazität einer Speicherseite entsprechende Anzahl, so wird die Region auf einer Speicherseite abgespeichert, andernfalls wird eine Spaltadresse für die Region derart ausgewählt, daß durch Spalten entlang der Spaltadresse eine erste Teilregion und eine zweite Teilregion erzeugt werden, die jeweils etwa die Hälfte der in der zu spaltenden Region enthaltenen Datenobjekte enthalten, woraufhin die entstandenen Teilregionen auf jeweils einer Speicherseite abgespeichert werden.

Vorteilhafterweise erfolgt auch bei diesem Verfahren das Auffinden der Speicherseite mittels in einem B-Baum, B*-Baum oder Präfix-B-Baum gespeicherter, den Speicherseiten zugeordneter Adressen und Zeiger.

Zur weiteren Lösung der der Erfindung zugrundeliegenden Aufgabe wird ein Verfahren zum Durchführen einer Datenanfrage auf der Grundlage eines gegebenen n-dimensionalen Anfragebereichs mit den Merkmalen des Anspruches 17 vorgeschlagen. Danach werden die Koordinaten des niedrigsten und des höchsten Schnittpunktes des gegebenen Anfragebereichs mit dem n-dimensionalen Datenbestand sowie diejenige Region bestimmt, in welcher der niedrigste Schnittpunkt liegt. Danach wird die Speicherseite aufgefunden, auf der die ermittelte Region abgespeichert ist, und es werden sämtliche auf dieser Speicherseite abgespeicherte Datenobjekte ermittelt, die eine Schnittmenge mit dem Anfragebereich bilden. Die ermittelten Datenobjekte werden daraufhin ausgegeben. Dann wird der in der Abfolge letzte Subwürfel der ermittelten Region bestimmt, der den Anfragebereich schneidet und die Datenanfrage wird beendet, falls der höchste Schnittpunkt des Anfragebereichs in diesem Subwürfel liegt. Andernfalls wird der nächste Subwürfel derselben Ebene und desselben übergeordneten Würfels ermittelt, der den Anfragebereich schneidet, und es werden die Koordinaten des niedrigsten Schnittpunktes des Anfragebereichs mit dem neu ermittelten Subwürfel bestimmt, woraufhin das Verfahren beim Bestimmen derjenigen Region, in welcher der niedrigste Schnittpunkt liegt, weitergeführt wird, falls ein Subwürfel ermittelt wurde. Andernfalls wird der nächste Subwürfel der Ebene des übergeordneten Würfels ermittelt, der den Anfragebereich schneidet und die Ermittlung des nächsten Subwürfels derselben Ebene und desselben übergeordneten Würfels, der den Anfragebereich schneidet, wird mit den Subwürfeln des neu ermittelten Würfels durchgeführt. Wird kein Subwürfel der Ebene des übergeordneten Würfels ermittelt, übernimmt der übergeordnete Würfel die Rolle des Subwürfels und es wird dann der nächste Subwürfel dieser Ebene und desselben übergeordneten Würfels ermittelt, der den Anfragebereich schneidet. So werden erfindungsgemäß sukzessive die Subwürfel aller relevanten übergeordneten Würfel und wiederum derer übergeordneter Würfel bezüglich Schnittmengen von Datenobjekten mit dem Anfragebereich überprüft.

Die Erfindung ist in den Zeichnungen anhand von Ausführungsbeispielen dargestellt und wird im folgenden unter Bezugnahme auf die Zeichnungen ausführlich erläutert.

Fig. 1 zeigt einen 2-dimensionalen Würfel, in dem ein nicht näher dargestellter 2-dimensionaler Datenbestand liegt und der in vier gleichgroße Subwürfel unterteilt ist.

Fig. 2 zeigt einen 3-dimensionalen Würfel, in dem ein nicht näher dargestellter 3-dimensionaler Datenbestand liegt und der in acht gleichgroße Subwürfel unterteilt ist.

Fig. 3.1 bis 3.4. dienen zur Veranschaulichung der Adreß-Zuweisung von Subwürfeln in einem 2-dimensionalen Würfel.

Fig. 4 dient zur Veranschaulichung der Adreß-Zuweisung von Subwürfeln in einem 3-dimensionalen Würfel.

Fig. 5 veranschaulicht die Abspeicherung von Adressen und Zeigern.

Fig. 6.1 und 6.2 veranschaulichen die Modifizierung und Abspeicherung von Adressen und Zeigern bei der Aufspaltung einer Region beim Einfügen von Datenobjekten.

Fig. 7 zeigt einen in eine Vielzahl von Regionen aufgeteilten 2-dimensionalen Würfel mit einem nicht näher dargestellten Datenbestand.

Fig. 8 zeigt einen Anfragebereich für den 2-dimensionalen Fall.

Fig. 9 zeigt den Abfragebereich der Fig. 3 mit einem in dem Abfragebereich liegenden Subwürfel.

Fig. 10 zeigt den Abfragebereich der Fig. 8 und 9 mit mehreren in dem Anfragebereich liegenden und den Anfragebereich schneidenden Subwürfeln.

Fig. 11 zeigt einen Datenbestand am Beispiel eines ausgedehnten Objekts.

Zur Verwaltung eines n-dimensionalen Datenbestands wird erfindungsgemäß davon ausgegangen, daß der Datenraum, in dem sich die zu verwaltenden Datenobjekte befinden, ein n-dimensionaler Würfel ist oder von einem solchen Würfel eingeschlossen wird, wobei n eine beliebige natürliche Zahl mit $n > 1$ ist. Dieser Würfel wird im gebender Würfel genannt.

Der umgebende Würfel wird unterteilt in 2^n Subwürfel gleicher Größe, indem jede Dimension des Würfels halbiert wird. Diese Subwürfel werden in einer festzulegenden Reihenfolge von 1 bis 2^n durchnummeriert. Fig. 1 zeigt einen 2-dimensionalen Würfel, also ein Quadrat, für den Fall eines 2-dimensionalen Datenraums, unterteilt in $2^2 = 4$ gleichgroße Subwürfel, die links oben beginnend von links nach rechts und von oben nach unten durchnumeriert sind.

Fig. 2 zeigt einen umgebenden Würfel eines 3-dimensionalen Datenraums mit $2^3 = 8$ gleichgroßen Subwürfeln die ebenfalls durchnumeriert sind, und zwar hinten links beginnend von links nach rechts, von oben nach unten und von hinten nach vorn. Demgemäß kommt der Subwürfel mit der Nummer 3 in der Zeichnung der Fig. 2 links hinten unten zu liegen und ist nicht sichtbar.

Jeder der in den Fig. 1 und 2 gezeigten Subwürfel des Ausgangswürfels kann nach demselben Verfahren wiederum in 2^n Subwürfel mit der Numerierung von 1 bis 2^n unterteilt werden und diese Unterteilung kann rekursiv (bzw. iterativ) beliebig oft fortgesetzt werden. In der praktischen Anwendung der Erfindung wird die Unterteilung solange durchgeführt, bis durch zusammenliegende Subwürfel Regionen gebildet werden können, deren Anzahl von Datenobjekten auf einer Speicherseite gegebener Speicherkapazität speicherbar sind.

Hat der Ausgangswürfel eine Seitenlänge l, so haben die Subwürfel nach s Unterteilungen noch die Seitenlänge $(1/2^s)*l$. Zur Identifizierung der Würfel werden diese entsprechend der jeweiligen Unterteilung in eine Stufe s eingeordnet. Demgemäß hat der Ausgangswürfel die Stufe 0 die in den Fig. 1 und 2 dargestellten Subwürfel haben die Stufe 1 usw.

Eine Area A ist nun ein spezieller Unterraum des Ausgangswürfels, der wie folgt entsteht:

- auf der Stufe 1 gehören die ersten a_1 Subwürfel vollständig zu der Area A, wobei $0 <= a_1 < 2^n$ ist
- auf der Stufe 2 gehören die ersten a_2 Subwürfel des Subwürfels $a_1 + 1$ der ersten Stufe zu der Area A
- usw. bis zur Stufe i, wo die ersten a_i Subwürfel des Subwürfels $a_{i-1} + 1$ der i-ten Stufe zu der Area A gehören.

Eine auf diese Weise definierte Area A ist eindeutig beschrieben durch die Folge von Zahlen $a_1\ a_2\ a_3 \ldots a_i$. Diese Zahlenfolge wird die eindeutige Adresse alpha (A) der Area A genannt. Dies ist anhand der Fig. 3.1 bis 3.4 für den 2-dimensionalen Fall, also $n = 2$, veranschaulicht.

Fig. 3.1 zeigt einen 2-dimensionalen Würfel, der in vier gleichgroße Subwürfel unterteilt ist, die jeweils wiederum zweimal unterteilt sind. Die in der Fig. 3.1 grau unterlegte Fläche bildet eine Area A, welche die Adresse alpha (A) = 03 hat, weil der erste 2-dimensionale Subwürfel auf der Stufe 1 nicht vollständig zu A gehört, was durch die Ziffer 0 an der ersten Stelle der Adresse angegeben ist. Von diesem Subwürfel gehören dann aber die ersten 3 Subwürfel der Stufe 2 zu A, was durch die Ziffer 3 an der zweiten Stelle der Adresse angegeben wird.

Fig. 3.2 zeigt den Würfel der Fig. 3.1 mit einer anderen grau unterlegten Fläche, die die Area B bildet. Diese hat die Adresse alpha (B) = 132, da der erste Subwürfel komplett in der Area B enthalten ist (Ziffer 1 an der ersten Stelle), von dem zweiten Subwürfel jedoch nur die ersten drei Subwürfel der Stufe 2 enthalten sind (Ziffer 3 an der zweiten Stelle) und von dem vierten Subwürfel der Stufe 2 wiederum nur die ersten beiden Subwürfel der Stufe 3 enthalten sind (Ziffer 2 an der dritten Stelle).

Fig. 3.3 zeigt wiederum den 2-dimensionalen Subwürfel der Fig. 3.1 und 3.2 mit einer unterschiedlichen grau unterlegten Fläche, die die Area C bildet, welche die Adresse alpha (C) = 2331 hat.

In Fig. 3.4 bildet der gesamte (grau unterlegte) 2-dimensionale Würfel eine Area D mit der Adresse alpha (D) = 4. Diesem Spezialfall wird als Adresse beispielsweise epsilon zugeordnet (alpha (D) = epsilon).

Zur weiteren Veranschaulichung der Adreßzuweisung ist in Fig. 4 ein eine Area E bildender Ausschnitt aus einem 3-dimensionalen Datenbestands-Würfel dargestellt, der die Adresse alpha (E) = 541 hat.

Die Subwürfel, die noch zu einer Area gehören, werden auf jeder Unterteilungsstufe exponentiell um den Faktor 2^n kleiner. Dadurch bleiben die Adressen sehr kurz. Bei einer Umsetzung beispielsweise einer Landkarte des Bundeslands Bayern hat eine Area, deren kleinster Subwürfel 8 × 8 Meter groß ist, eine Adresse von etwa 32 Bits Länge.

Die beschriebenen Areas sind streng linear nach ihrem mengentheoretischen Enthaltensein geordnet: Für eine Area A, die raummäßig in einer Area B enthalten ist, kann man schreiben:

A enthalten in B

Des weiteren sind die zu den Areas gehörigen Adressen lexikographisch wie Wörter über einem Alphabet geordnet. Ist beispielsweise eine Adresse α kleiner als Adresse β, dann kann die bezeichnet werden durch

$\alpha \prec \beta$

So gilt beispielsweise

132 ≺ 2331 und 2331 ≺ 32

Die vorstehend beschriebenen Areas und Adressen sind nun so aufgebaut, daß folgender Zusammenhang gilt:

Area(α) enthalten in Area(β) genau dann wenn α ≺ β

Da die Areas wie erläutert linear geordnet sind, kann immer die Differenz zwischen der größeren und der kleineren Area gebildet werden. ist die Adresse α kleiner als die Adresse β, also α ≺ β, dann wird eine Region reg (α, β) als die Differenz zwischen Area(β) und Area(α) definiert. Dies ist gleichbedeutend mit:

reg(α,β) = Area(β) — Area(α)

Regionen haben die Eigenschaft, daß sie im gegebenen n-dimensionalen Raum nach sehr speziellen Mustern geclustert sind. Fig. 7 zeigt ein Beispiel einer derartigen Clusterung eines 2-dimensionalen Würfels in mehrere Regionen. So umfaßt die erste, mit 01 bezeichnete Region exakt den ersten Subwürfel des ersten Subwürfels des Gesamtwürfels. Die zweite Region mit 023 bezeichnet, umfaßt den kompletten zweiten Subwürfel des ersten Subwürfels und die drei ersten Subwürfel des folgenden dritten Subwürfels. In der Fig. 7 ist diese Region weiß dargestellt. Die Region beginnt somit nach dem Subwürfel mit der Adresse 01 und endet mit dem Subwürfel der Adresse 023. Durch die Angabe dieser beiden Adressen ist somit gemäß der vorstehend erläuterten Vorschrift, nämlich reg (01, 023) = Area (023) — Area (01), eindeutig definiert.

Die weiteren in dem Subwürfel der Fig. 7 eingezeichneten Regionen werden entsprechend unter Verwendung der in den einzelnen Regionen angegebenen Adressen gebildet. Dabei kann es durchaus vorkommen, daß Subwürfel, die eine zusammenhängende Region bilden, aufgrund der Nummerierungs- und Darstellungsweise als nicht zusammenhängend erscheinen. Dies ist genau dann der Fall, wenn eine Region sich aus Subwürfeln zusammensetzt, bei denen in der Numerierung ein Sprung von 2 nach 3 oder von 4 nach 1 vorliegt. In der Fig. 7 ist dies beispielsweise bei der schon erläuterten weiß eingezeichneten Region reg (01, 023) der Fall, da dort der Subwürfel 02 und Teile des Subwürfels 03 eine Region bilden, durch den Übergang der Numerierung von 02 nach 03 jedoch in der Darstellung ein Sprung vorliegt. Dies ist u. a. auch in der sich an die Region reg (01, 023) anschließenden Region reg (023, 101) der Fall, die sich von dem Subwürfel 04 in den Subwürfel 11 erstreckt.

Die beschriebenen Regionen eines n-dimensionalen Würfels spielen eine zentrale Rolle bei der Speicherung von Objekten auf einem insbesondere peripheren Computerspeicher. Diese Speicher sind unterteilt in sogenannte Seiten, deren Inhalt bei einem Ein/Ausgabevorgang mit einem Speicherzugriff in den Arbeitsspeicher des Rechners geholt bzw. von dort wieder in den peripheren Computerspeicher zurückgeschrieben wird. Die Regionen werden im folgenden so konstruiert, daß die abzuspeichernden Datenobjekte, die in einer Region liegen bzw. sie schneiden, auf einer Seite des peripheren Computerspeichers gespeichert werden können. Die zur Region reg(α, β) zugehörige Speicherseite kann beispielsweise mit page (α, β) bezeichnet werden.

Bei allen Computeranwendungen spielt die Auflösung des Raumes, d. h. die kleinsten noch unterscheidbaren Raumelemente eine wesentliche Rolle. Sie werden im 2-dimensionalen Fall auch pixel (für picture element) und im n-dimensionalen Fall voxel (für volume element) genannt. Die Anzahl der Elemente, die pro Dimension unterschieden werden können, kann mit pix bezeichnet werden. Sind $(x_1, x_2, \ldots x_n)$ die kartesischen Koordinaten eines Punkts im n-dimensionalen Raum, dann gilt $0 <= x_i < pix$ für $i = 1, 2, \ldots, n$.

Die Area, deren letzter Subwürfel gerade der Punkt $(x_1, x_2, \ldots x_n)$ ist, ist eindeutig definiert und hat eine bestimmte Adresse, die sich aus $(x_1, x_2, \ldots x_n)$ leicht und eindeutig berechnen läßt. Wir bezeichnen diese Funktion bzw. Berechnungsvorschrift mit alpha $(x_1, x_2, \ldots x_n)$. Umgekehrt lassen sich aus der Adresse α einer Area die kartesischen Koordinaten des letzten Punktes berechnen, der noch zu dieser Area gehört. Wir bezeichnen diese Funktion mit cart(α). alpha und cart sind inverse Funktionen zueinander, d. h. es gilt:

cart(alpha$(x_1, x_2, \ldots x_n)$) = $(x_1, x_2, \ldots x_n)$
alpha(cart(α)) = α.

Wie beschrieben wird bei der erfindungsgemäßen Datenverwaltung ein n-dimensionaler Raum in Form eines Würfels durch eine Menge von Areas vollständig in Regionen partitioniert, wobei die Adressen der Areas lexikographisch sortiert werden und eine Region gerade durch die Differenz zweier aufeinanderfolgender Areas bzw. durch deren Adressen definiert wird. Die kleinst mögliche Area ist das kleinste Pixel des "Datenuniversums" und hat die Adresse 00 ... 01. Diese Adresse wird hier mit sigma bezeichnet. Die größte Adresse ist 4 im 2-dimensionalen Fall und 2^n im n-dimensionalen Fall und wird hier, wie vorstehend schon ausgeführt mit epsiion bezeichnet.

Die sortierten Adressen der Areas werden erfindungsgemäß in einem konventionellen B-Baum, B*-Baum oder Präfix-B-Baum abgelegt. Vorteilhafterweise wird im B-Baum zwischen zwei aufeinanderfolgenden Adressen α_{i-1} und α_i, welche genau die Region reg (α_{i-1}, α_i) definieren, ein Zeiger (auch Pointer oder Verweis genannt) auf diejenige Seite des peripheren Speichers abgelegt, auf der die Datenobjekte der Region reg (α_{i-1}, α_i) gespeichert sind. Dieser Zeiger wird mit p_i bezeichnet.

Fig. 5 veranschaulicht eine derartige Abspeicherung, wobei in der in der Darstellung der Fig. 5 oberen Ebene alternierend jeweils eine Adresse und ein Zeiger abgespeichert sind. Durch die beiden jeweils einem Zeiger zugeordneten Adressen sind die Grenzen einer Region gegeben, auf deren Datenobjekte der zwischen den beiden Adressen stehende Zeiger verweist. So zeigt im Beispiel der Fig. 5 der Zeiger p_i durch den eingezeichneten Pfeil auf eine Speicherseite, in welcher die Datenobjekte (hier die Identifikatoren der Datenobjekte) der durch die jeweils links und rechts des Zeigers stehenden Adressen gebildete Region reg (α_{i-1}, α_i) liegen.

Der hier beschriebene Fall betrifft einen B*-Baum für Adressen, bei dem die Zeiger p_i auf sogenannte Blattseiten zeigen. Auf diesen Blattseiten stehen dann die Datenobjekte selbst oder deren Identifikatoren, wobei dann die Datenobjekte selbst noch einmal auf weitere Seiten ausgelagert sind und über ihre Identifikatoren im peripheren Speicher gefunden werden können. Die vorstehend beschriebene erfindungsgemäße Datenstruktur wird im folgenden als FB-Baum bezeichnet.

Nachfolgend wird das erfindungsgemäße Verfahren am Beispiel der Verwaltung von Punktobjekten im n-dimensionalen Raum beschrieben. Ein Punktobjekt P ist durch seine kartesischen Koordinaten $(x_1, x_2, \ldots x_n)$ gegeben. Daraus wird die Adresse β = alpha $(x_1, x_2, \ldots x_n)$ berechnet. Der Punkt P liegt in der eindeutig definierten Region reg (α_{j-1}, a_j) mit der Eigenschaft daß

$$\alpha_{j-1} \prec \beta \prec = \alpha_j$$

Diese Region wird durch eine Baumsuche im FB-Baum bestimmt, wo der Verweis P_j auf die Seite page (α_{j-1}, a_j) gefunden wird. Der Punkt P, d. h. sein Identifikator zusammen mit seinen Koordinaten $(x_1, x_2, \ldots x_n)$ wird dann auf der Seite page (α_{j-1}, α_j) des Peripheriespeichers abgespeichert. Alternativ dazu kann nur der Identifikator des Punktes auf page (α_{j-1}, α_j) gespeichert werden und der Punkt selbst, d. h. seine Koordinaten und sonstige Informationen über ihn, wird nochmals auf eine andere Seite ausgelagert.

Die Seiten des peripheren Speichers haben nur eine bestimmte vorgegebene Speicherkapazität und können deshalb nur eine bestimmte Anzahl M von Objekten aufnehmen. Sobald in eine Region weitere Objekte eingefügt werden sollen, die zur Region gehörende Seite aber keine Objekte mehr aufnehmen kann, muß der Inhalt der Seite auf zwei Seiten aufgeteilt werden, und die Region muß dementsprechend in zwei Regionen gespalten werden. Nachfolgend wir zunächst die Spaltung der Region beschrieben: Sei die Region reg (α_{j-1}, α_j) und die zugehörige Seite page (α_{j-1}, α_j). Wegen der Definition einer Region gilt dann: $\alpha_{j-1} \prec \alpha_j$. Jetzt wird eine Spaltadresse β gewählt mit der Eigenschaft, daß β zwischen den beiden Arca-Adressen α_{j-1} und α_j liegt und die Region reg $(\alpha_{j-1}, \alpha_j$ etwa in der Mitte spaltet, d. h. daß etwa die Hälfte der Objekte in der Region reg (α_{j-1}, β), die andere Hälfte in der Region reg (β, α_j) zu liegen kommen. Dann enthalten die beiden Regionen reg (α_{j-1}, β) und reg (β, α_j) jeweils weniger als 1/2 M + eps Objekte, wobei eps eine kleine vorgegebene Zahl ist und z. B. etwa 1/10 von M sein könnte.

Anschließend werden die Objekte von der Seite page (α_{j-1}, α_j) auf die beiden Seiten page (α_{j-1}, β) und page (β, α_j) aufgeteilt wobei eine der beiden Seiten mit der ursprünglichen Seite page (α_{j-1}, α_j) identisch sein kann.

Bei der Spaltung der Region reg (α_{j-1}, α_j) wird dabei die aus den Fig. 6.1 und 6.2 ersichtliche Modifikation des FB-Baums durchgeführt. Fig. 6.1 zeigt die Ausgangsstruktur des FB-Baums mit einer in der Zeichnungsdarstellung oberen Ebene, die die abgespeicherten Adressen α_{j-1}, α_j und α_{j+1} sowie die zwischen diesen Adressen liegenden Zeiger p_j und p_{j+1} enthält. Dabei verweist der Zeiger p_j auf die Seite page (α_{j-1}, α_j) und der Zeiger p_{j+1} auf die Seite page (α_{j-1}, α_j). Nach der Spaltung der Seite page (α_{j-1}, α_j) liegt die aus der Fig. 6.2 ersichtliche Baumstruktur vor, wobei in der in der Zeichnungsdarstellung oberen Ebene zwischen dem Zeiger p_j und der Adresse α_j die der Spaltadresse entsprechende Adresse β und der auf die neue Speicherseite page (β, α_j) verweisende Zeiger p' eingefügt wurden. Der bisherige Zeiger p_j verweist nunmehr auf die modifizierte Seite page (α_{j-1}, β) und der Zeiger p_{j+1} verweist unverändert auf die ebenfalls unveränderte Seite page (α_j, α_{j+1}).

Durch die Einfügung von β und p' in den übergeordneten Knoten muß möglicherweise die übergeordnete Seite ebenfalls gespalten werden, wenn dort durch diese Einfügung zuviel Adreß- und Zeigerdaten vorhanden sind. Diese Spaltvorgänge verlaufen dann aber genau wie bei den aus dem Stand der Technik bekannten B-Bäumen üblich. Durch diese wiederholten Spaltvorgänge beim Einfügen von Objekten in das vorhandene Datenuniversum entstehen die FB-Bäume, die ein ganz ähnliches Wachstum zeigen wie die bekannten B-Bäume.

Zur weiteren Erläuterung des Verfahrens wird nachfolgend die Löschung von Datenobjekten aus dem vorhandenen Datenuniversum und eine Anpassung der Regionen nach erfolgter Löschung beschrieben.

Soll ein Punkt-Objekt $(x_1, x_2, \ldots x_n)$ wieder gelöscht werden so werden zunächst wie beim Einfügevorgang die Region, in der der Punkt liegt, und die zugehörige Speicherseite bestimmt. Der Punkt wird von der Seite gelöscht und verschwindet somit auch aus der Region. Sinkt dadurch die Anzahl der Objekte in der Seite unter 1/2 M — eps, so wird die Region mit einer der beiden Nachbarregionen verschmolzen. Wenn die so entstehende Region zu viele Objekte enthält, wird sie wieder wie schon vorher beschrieben in der Mitte gespalten.

Beispielhaft: könnte die Region reg (α_{i-1}, a_i), falls sie nach einem Löschvorgang zu wenige Objekte enthält, mit der Region reg (α_i, α_{i+1}) verschmolzen werden zur Region reg $(\alpha_{i-1}, \alpha_{i+1})$. Falls reg $(\alpha_{i-1}, \alpha_{i+1})$ dann zu viele Objekte enthält, wird sie wieder gespalten in reg (α_{i-1}, β) und reg (β, α_{i+1}), wobei natürlich β geeignet gewählt wird und gelten muß

$$\alpha_{i-1} \prec \beta \prec \alpha_{i+1}$$

wobei außerdem $\alpha_i \prec \beta$ sein wird, damit mehr Objekte in der ersten Region zu liegen kommen. Beim Löschen eines Objektes aus der Region reg (α_{i-1}, α_i) ergeben sich somit die folgenden 3 Fälle:

Fall 1: reg (α_{i-1}, α_i) hat nach dem Löschen noch mindestens 1/2 M — eps Objekte. Dann bleiben die Region und die zugehörige Seite erhalten.

Fall 2: reg (α_{i-1}, α_i) kann mit einer der beiden Nachbarregionen reg $(\alpha_{i-2}, \alpha_{i-1})$ oder reg (α_i, α_{i+1}) verschmolzen werden zu der neuen Region reg (α_{i-2}, α_i) bzw. reg $(\alpha_{i-1}, \alpha_{i+1})$.

Fall 3: Die Region reg (α_{i-1}, α_i) wird wie im Fall 2 zunächst mit einer Nachbarregion verschmolzen, muß aber anschließend wieder gespalten werden und es entstehen die beiden Regionen reg (α_{i-2}, β) und reg (β, α_i) bzw. reg (α_{i-1}, β) und reg (β, α_{i+1}).

Durch solche Löschvorgänge können benachbarte Regionen irgendwann wieder endgültig verschmolzen werden, so daß der FB-Baum wieder schrumpft und genau das umgekehrte Verhalten zeigt wie beim Spalten von Regionen und Seiten und dem dadurch hervorgerufenen Wachstum. Sind schließlich alle Objekte aus dem Universum gelöscht, so ist der FB-Baum wieder leer geworden.

Zur weiteren Erläuterung des erfindungsgemäßen Verfahrens wird nachfolgend die Beantwortung von Punktanfragen erläutert.

Bei einer Punktanfrage werden die kartesischen Koordinaten $(y_1 \ldots y_n)$ des gesuchten Punktes P angegeben, über den dann Zusatzinformationen wie beispielsweise Höhe oder Temperatur oder Börsenwert oder dergleichen in Erfahrung gebracht werden sollen. Diese Zusatzinformationen sind mit dem Punktobjekt selbst abgespeichert.

Zunächst wird aus den gegebenen kartesischen Koordinaten $(y_1, \ldots, y_n)$ die Adresse pp des Punktes P berechnet. Der Punkt liegt in der eindeutig bestimmten Region reg (α_{j-1}, α_j) mit der Eigenschaft

$$\alpha_{j-1} \prec pp \prec = \alpha_j.$$

Diese Region und die zu dieser Region gehörende Seite page (α_{j-1}, α_j) wird mittels einer Suche im FB-Baum und über den dort abgelegten Zeiger p_j gefunden und geholt. Auf der Seite page (α_{j-1}, α_j) befindet sich dann die vollständige gewünschte Information über den Punkt P (und natürlich über weitere Punkte und Objekte, die zu dieser Region gehören).

Da ein erfindungsgemäßer FB-Baum genau wie ein B-Baum bzgl. der Höhe balanciert ist, kann die Baumsuche und somit das Auffinden des Punktes P in einer Zeit $O(\log_k N)$ durchgeführt werden.

Eine fundamentale Anfrageart bei allen Datenbanksystemen sind sogenannte Bereichsanfragen, bei welchen bezüglich jeder Dimension ein Intervall vorgegeben wird. Keine Intervallangabe bezüglich einer Dimension wird dabei als das Intervall (– unendlich, + unendlich) betrachtet. Durch das Produkt dieser Intervalle wird ein n-dimensionaler Quader bestimmt, der den Anfragebereich darstellt. Im folgenden wird dieser Anfragebereich Query-Box q genannt.

Fig. 8 zeigt beispielhaft eine Query-Box q für den 2-dimensionalen Fall, in welchem der n-dimensionale Quader ein Rechteck ist. Der in der Fig. 8 dargestellte Anfragebereich der Query-Box q ist gegeben durch die Werte (ql_1, ql_2) für den niedrigsten Wert (1 für low) und (qh_1, qh_2) für den höchsten Punkt (h für high).

Die Antwort auf eine Bereichsanfrage ist die Menge derjenigen Punkte oder Objekte, die in der Query-Box q liegen oder diese schneiden.

Im allgemeinen n-dimensionalen Fall ist die Query-Box gegeben durch die $2*n$ Werte ql_i und qh_i mit $i = 1, 2, \ldots, n$, wobei natürlich gemäß vorstehend erläutertem Beispiel der Fig. 8 $ql_i < qh_i$ gilt (für den Fall $ql_i = qh_i$ für alle i ergibt sich der Spezialfall der Punktanfrage, der schon behandelt wurde).

Der kleinste Punkt der Query-Box q hat somit die kartesischen Koordinaten $(ql_1, ql_2, \ldots, ql_n)$ und liegt in einer genau definierten Region reg (α_{j-1}, α_j). Zum Auffinden dieser Region wird zunächst die Adresse

$$\lambda = \text{alpha}(ql_1, ql_2, \ldots, ql_n)$$

des kleinsten Punktes von q berechnet. Dabei handelt es sich um hauptspeicherinterne Berechnungen, die keine Zugriffe zu peripheren Speichern erfordern. Der Rechenaufwand ist aus diesem Grunde vernachlässigbar klein.

Dann wird die Region reg (α_{j-1}, α_j) mit der Eigenschaft

$$\alpha_{j-1} \prec \lambda \prec = \alpha_j$$

bestimmt. Das erfordert eine Suche im FB-Baum mit einem Aufwand $O(\log_k N)$. Dabei können auch $O(\log_k N)$ Plattenzugriffe erforderlich werden. Die letzte Seite dieser Baumsuche ist die Seite page (α_{j-1}, α_j), die die Identifikatoren aller Datenobjekte bzw. die vollständigen Datenobjekte selbst enthält, die in der Region reg (α_{j-1}, α_j) liegen oder diese Region schneiden.

Für diese Datenobjekte wird nun einzeln bestimmt, ob sie die Query-Box q schneiden oder nicht. Es ist zu beachten, daß die Datenobjekte q nur schneiden können, wenn ihre zugehörige Region die Query-Box q schneidet (hierbei handelt es sich um eine notwendige, aber nicht hinreichende Bedingung). So wird zunächst ein Teil der Datenobjekte in der Query-Box q gefunden.

Die gefundene Region reg (α_{j-1}, α_j) ist wie vorstehend beschrieben aus Subwürfeln aufgebaut, deren Adressen geordnet sind. Als Beispiel soll hier die Region reg (023, 101) aus Fig. 7 dienen, die aus den Würfeln 024, 04, 101 in dieser Reihenfolge besteht. Wenn eine Region den Anfragebereich schneidet, müssen natürlich nicht alle Subwürfel dieser Region den Anfragebereich ebenfalls schneiden.

Die Adresse des letzten Subwürfels der Region reg (α_{j-1}, α_j), der den Anfragebereich schneidet, sei β. Außerdem habe β die Form $\beta' l$ wobei l der Index von β auf der Stufe ist, auf der sich β befindet. Beispielsweise gilt für den Würfel 024 der Region reg (023 101) der Fig. 7:

für 024 auf der Stufe 3 ist $l = 4$.

$\beta = 024$ läßt sich somit in der Form $\beta' l$ mit $\beta' = 02$ und $l = 4$ darstellen, wobei die Anzahl der Ziffern in der Adreßdarstellung die Stufe wiedergibt.

Dasselbe gilt für die weiteren in der genannten Region reg (023, 101) vorhandenen Würfel 04 und 101:

für 04 auf der Stufe 2 ist $l = 4$

für 101 auf der Stufe 3 ist l = 1.

Es ist zu beachten, daß l = 0 nicht vorkommen kann, da nach der erfindungsgemäßen Konstruktion der Adressen keine Adresse mit 0 endet.

Nachdem die Region reg (α_{j-1}, α_j) wie vorstehend beschrieben bei der Bereichsanfrage abgearbeitet ist, muß nun die nächste Region gefunden werden, die den Anfragebereich schneidet. Dazu wird die Lage von Würfel β im Bezug zu der Query-Box q und zu seinem übergeordneten Würfel, in dem β selbst enthalten ist, betrachtet. Dies ist beispielhaft in der Darstellung der Fig. 9 gezeigt, in welcher die Query-Box q der Fig. 8 mit darin liegendem Würfel β dargestellt ist, wobei die weiteren Würfel, die zu dem übergeordneten Würfel des Würfels β gehören, punktiert eingezeichnet sind. Diese punktiert eingezeichneten Subwürfel der gleichen Stufe, die zu einem selben übergeordneten Würfel gehören, werden hier als Brüder bezeichnet. Alle Subwürfel der Stufe s eines übergeordneten Würfels der Stufe s-l sind somit Brüder. Ein Subwürfel der gleichen Stufe s mit einem kleineren Index l ist damit ein kleinerer Bruder, ein Subwürfel derselben Stufe s mit einem größeren Index l dabei ein größerer Bruder. In dem in Fig. 7 dargestellten Beispiel ist der große Bruder des Subwürfels 023 der Subwürfel 024, wobei diese beiden Brüder nicht in derselben Region liegen.

Bei der Weiterführung der Bereichsanfrage sind somit die nächsten Datenobjekte, die bisher in der schon abgearbeiteten Region reg (α_{j-1}, α_j) noch nicht gefunden wurden und in der Query-Box q liegen, in einem größeren Bruder von β oder im Vater von β oder in einem anderen Vorfahren von β enthalten.

Fig. 10 zeigt beispielhaft einige mögliche Situationen für einen Würfel β der Form β'2 für den 2-dimensionalen Fall.

Falls sich kein größerer Bruder des Würfels β mit der Query-Box q schneidet (letzter Fall in Fig. 10), dann enthält auch der Vater des Würfels β keine Objekte mehr, die noch nicht gefunden sind (durch vorherige Abarbeitung der kleineren Brüder), aber in der Query-Box q liegen könnten. Es müssen deshalb die größeren Brüder des Vaters des Würfels β dahingehend untersucht werden, ob sie die Query-Box q schneiden. Falls nicht, muß nach einer analogen Überlegung zum Großvater des Würfels β übergegangen und dessen größere Brüder auf Überschneidung mit der Query-Box q geprüft werden etc. Auf diese Weise wird schließlich das ganze Datenuniversum abgedeckt und alle Objekte in der Query-Box q werden gefunden.

Man beachte: Wenn sich der Würfel β auf der Stufe s befindet, dann können wir höchstens s-mal zum Vater übergehen und jeweils die größeren Brüder (davon gibt es höchstens $2^n - 1$) auf Überschneidung mit der Query-Box q prüfen. Außerdem ist s < = ld (pix) und beim Übergang zum Vaterknoten sowie bei der Prüfung der Überschneidung der jeweils größeren Brüder mit der Query-Box q werden nur hauptspeicherinterne Berechnungen durchgeführt; es sind also keine Ein/Ausgabevorgänge oder Plattenzugriffe nötig. Deshalb ist dieses Verfahren, den nächsten Würfel zu finden, der die Query-Box q schneidet, extrem schnell und kann in der Bilanz für die Gesamtzeit bei der Beantwortung einer Bereichsanfrage vernachlässigt werden.

Sobald nach diesem Verfahren der erste Würfel bestimmt wurde, der die Query-Box q schneidet, werden die kartesischen Koordinaten des kleinsten Pixels im Durchschnitt mit der Query-Box q (in Fig. 10 die kleinen schwarzen Quadrate) berechnet, die sich wie folgt ergeben:
Ein Würfel habe bzgl. Dimension i die Ausdehnung von xl_i (kleinste Koordinate) bis xh_i (größte Koordinate) in Anlehnung an die Bezeichnungen für die Werte ql bzw. qh.

Die Bedingung dafür, daß dieser Würfel die Query-Box q nicht schneidet, ist:

es existiert i: $xh_i < ql_i$ or $xl_i > qh_i$

Die Bedingung dafür, daß dieser Würfel die Query-Box q schneidet, ist die Negation obiger Formel:

not: es existiert i: $xh_i < ql_i$ or $xl_i > qh_i$

bzw. nach den Gesetzen der mathematischen Logik:

for all i: $xh_i > = ql_i$ and $xl_i < = qh_i$

Dann ergeben sich die Koordinaten des kleinsten Schnittpunktes sp mit der Query-Box q wie folgt bzgl. der i-ten Dimension:

if $xl_i > = ql_i$ then $sp_i := xl_i$ else $sp_i := ql_i$

Der Schnittpunkt sp hat dann die kartesischen Koordinaten

$sp = (sp_1, sp_2, \ldots, sp_n)$

und seine Adresse ist:

sigma: = alpha $(sp_1, sp_2, \ldots, sp_n)$

Man beachte, daß bis zu diesem Schritt unser Verfahren zur Bestimmung von sp keine Ein/Ausgabevorgänge oder Plattenzugriffe benötigte.

Um die zu dem Schnittpunkt sp gehörende eindeutige Region zu finden ist nun eine Punktanfrage erforderlich mit der Adresse sigma, genau wie sie oben schon beschrieben wurde. Sie erfordert einen Zeitaufwand von der

Ordnung O ($\log_k N$), wie oben ebenfalls schon analysiert wurde.

Daraus folgt nun aber, daß bei der Beantwortung einer Bereichsanfrage nur Bearbeitungskosten für diejenigen Regionen anfallen, die die Query-Box q tatsächlich schneiden. Für jede solche Region sind die Kosten von der Ordnung O ($\log_k N$), insgesamt also $r*O(\log_k N)$ Kosten, wenn r Regionen die Query-Box q schneiden.

Nachfolgend wird das Verfahren zur Beantwortung von Bereichsanfragen für eine n-dimensionale Query-Box q mit Koordinaten ql_i und qh_i für $i = 1, 2, \ldots, n$ angegeben:

```
Initialize:
    sigma := alpha (ql1 , ql2 ,..., qln) ;

RegionLoop: begin co für jede Region, die q schneidet oc
    finde durch Baumsuche im FB-Baum die Region reg ( α(j-1) , αj),
        in der sigma liegt, d.h. mit α(j-1) << sigma <<= αj ;
    hole Seite page (α(j-1) , αj);

    ObjectLoop:    begin für alle Objekte Q auf Seite page (α(j-1) , αj)
prüfe:
        if Q intersects q then Ausgabe von Q als Teil der Antwort
    end ObjectLoop ;

    finde letzten Subcube mit Adresse β von reg (α(j-1) , αj),
        so daß Subcube (β) intersects q;

    if (qh1 , qh2 ,..., qhn) contained in Subcube (β) then co fertig oc goto
Exit else
    FatherLoop: begin co β habe die Form β = β'.i oc
        i := tail (β) ;

        BrotherLoop: for k := i+1 to 2^n
            do if Subcube (β'.k) intersects q then
                begin sp := smallest intersection with q;
                    sigma := alpha ( sp );
                    goto RegionLoop
                        co Schleife wird hier sicher verlassen,
                        weil q noch nicht abgearbeitet ist oc
                end
            od co für alle größeren Brüder von β ist Durchschnitt mit
q leer oc ;

        β := father (β);
        goto FatherLoop
    end FatherLoop;

end RegionLoop ;
Exit: co Ende des Programms oc
```

Nachfolgend wird das erfindungsgemäße Verfahren beispielhaft anhand der Verwaltung von allgemeinen ausgedehnten Objekten beschrieben.

Ein allgemeines ausgedehntes Objekt ist ein wichtiger Fall eines Datenobjekts. Dabei handelt es sich beispielsweise um einen See in einer geographischen Karte, wie er in Fig. 11 dargestellt ist.

Das ausgedehnte Objekt O wird zuerst durch einen der Dimension entsprechenden achsenparallelen Quader umgeben, der in seinen Abmessungen so klein gewählt wird, daß er das ausgedehnte Objekt gerade umgibt. In

dem Beispiel der Fig. 11 ist dies an dem dort abgebildeten See und einem den See gerade umgebenden 2-dimensionalen Quader bb (= Rechteck) dargestellt. In der Fachliteratur wird der das ausgedehnte Objekt umgebende Quader als Bounding Box bezeichnet.

Für ein ausgedehntes Objekt wird im FB-Baum lediglich der Identifikator Id(O) abgespeichert, wobei diese Abspeicherung des Identifikators im allgemeinen sogar mehrfach erfolgt, nämlich für jede Region, die das ausgedehnte Objekt schneidet. Das ausgedehnte Objekt selbst ist aus dem FB-Baum in einen anderen Speicherbereich oder in eine Datenbank ausgelagert.

Die Bounding Box für ein ausgedehntes Objekt O wird hier mit bb(O) bezeichnet. Der Identifikator Id(O) für das ausgedehnte Objekt O wird nun in dem FB-Baum mit jeder Region gespeichert, die das ausgedehnte Objekt O schneidet. Dabei ist zu beachten, daß das ausgedehnte Objekt O nur diejenigen Regionen schneiden kann, die auch von der Bounding Box bb geschnitten werden. Dies ist eine notwendige, jedoch keine hinreichende Bedingung, die eingesetzt wird, um die Algorithmen zur Umsetzung des erfindungsgemäßen Verfahrens wesentlich zu beschleunigen.

Beim Einfügen eines allgemeinen ausgedehnten Objekts O wird zuerst die zugehörige Bounding Box bb(O) berechnet. Dann wird das folgende Verfahren ausgeführt:

```
for all Regionen R, die bb(O) schneiden do
    if R intersects O then füge Id(O) in R ein
        co dies kann natürlich zu einer, i.a. sogar zu mehreren Spaltun-
        gen von R führen oc

Hinweis: Um die Regionen R zu finden, die bb(O) schneiden, behandelt
        man bb(O) genau wie eine Query Box q. Das führt zu folgen-
        dem detaillierten Verfahren:

Initialize:
        berechne bb(O);
        q := bb(O);
    sigma := alpha (ql1 , ql2 ,..., qln ) ;

RegionLoop: begin co für jede Region, die q schneidet oc
    finde durch Baumsuche im FB-Baum die Region reg (αj-1, αj),
        in der sigma liegt, d.h. mit αj-1 << sigma <<= αj ;
        hole Seite page (αj-1 , αj);
    if O intersects R then füge Id(O) in R ein, d.h.:
    if Anzahl der Objekte, die R schneiden, ist <= M
        then speichere Id(O) auf page (αj-1, αj)
        else spalte R und page (αj-1, αj)
        wie vorstehend zum Spalten von Regionen und Seiten
        beschrieben;

    finde letzten Subwürfel mit Adresse β von reg (αj-1, αj),
        so daß Subcube (β) intersects q;

if (qh1 , qh2 ,..., qhn) contained in Subcube (β) then co fertig oc goto Exit
    else
Father Loop: begin co β habe die Form β = β'.i oc
        i := tail (β) ;

        BrotherLoop: for k := i+1 to 2^n
            do if Subcube (β'.k) intersects q then
                begin sp := smallest intersection with q;
                    sigma := alpha ( sp );
                    goto RegionLoop
                        co Schleife wird hier sicher verlassen,
                        weil q noch nicht abgearbeitet ist oc
                end
            od co für alle größeren Brüder von β ist Durchschnitt mit
            q leer oc ;

        β := father (β);
        goto FatherLoop
    end FatherLoop;

end RegionLoop ;
Exit: co Ende des Programms oc
```

Zum Löschen eines ausgedehnten Objektes O wird wieder eine Bounding Box bb(O) verwendet, um alle Regionen zu finden die O schneiden könnten. Der Identifikator Id(O) ist in den Regionen enthalten und auf den zugehörigen Seiten gespeichert, die das ausgedehnte Objekt O tatsächlich schneiden, und wird daraus gelöscht. Dabei kann es wieder zum Verschmelzen von Regionen und zugehörigen Seiten kommen, wie es vorstehend schon beschrieben wurde.

Das erfindungsgemäße Datenbanksystem und die erfindungsgemäßen Verfahren zur Verwaltung multidimensionaler Daten gestatten somit einen schnellen und sicheren Zugriff auf Daten eines multidimensionalen Datenbestands, wobei die erfindungsgemäße Datenstruktur insbesondere ein dynamisches Ergänzen oder Ändern des multidimensionalen Datenbestands erlaubt. Erfindungsgemäß ist lediglich die Modifikation eines einzelnen Baumes für das Einfügen bzw. Löschen von Objekten notwendig.

Bei der Beantwortung von Anfragen zeigt das erfindungsgemäße FB-Baum-Verfahren folgende Leistungscharakteristik: Wir nehmen dazu an, daß p_i% der Werte des Datenbestandes bezüglich der i-ten Dimension in dem Anfrageintervall $(q_{li}; q_{hi}]$ der Query-Box q liegen, dann liegen in der Query-Box q

$$(p_1\% * p_2\% * \ldots * p_n\%) * N$$

Objekte. Da nicht alle Regionen, die q schneiden, vollständig innerhalb von q liegen, sondern darüber hinausragen können, müssen i.a. mehr Objekte vom peripheren Speicher geholt werden, als in der Query-Box liegen. Im Mittel sind das aber weniger als zweimal soviel Objekte, wie in q liegen, d. h. $2*(p_1\% * p_2\% * \ldots * p_n\%) * N$ Objekte.

Es liegt somit ein multiplikatives anstatt ein additives Verhalten (wie bei den aus dem Stand der Technik bekannten Verfahren) von Bruchteilen von N vor, was zu deutlichen Verbesserungen führt. Dies wird anhand eines einfachen Rechenbeispiels illustriert:

Seien $p_1 = 2\%$ $p_2 = 5\%$, $p_3 = 4\%$ $p_4 = 10\%$
dann ist die Summe der $p_i = 21\% = 21 * 10^{-2}$
und das Produkt der $p_i = 400 * 10^{-8} = 4 * 10^{-6}$.

Wenn das gesamte betrachtete Datenuniversum 10.000.000 Objekte enthält — für typische Datenbankanwendungen ein realistisches, eher kleines Datenuniversum — dann müssen mit dem derzeitigen Stand der Technik 2.100.000 Objekte vom peripheren Speicher geholt werden, mit dem neuen Verfahren der FB-Bäume aber nur $2 * 10^7 * 40^{-6} = 80$ Objekte, eine Verbesserung etwa um den Faktor 2500 gegenüber dem bekannten Stand der Technik.

Selbstverständlich ist die vorliegende Erfindung nicht auf die beschriebenen Ausführungsformen beschränkt, sondern es sind andere Ausgestaltungen möglich, die im Bereich des fachmännischen Könnens liegen. So ist beispielsweise die Art und Weise der Numerierung der Subwürfel und der Aufbau der Adressen von Subwürfeln und Regionen auch auf andere Weise möglich, ohne daß der Bereich der Erfindung verlassen wird.

Patentansprüche

1. Datenbanksystem mit einer Recheneinrichtung, einem Hauptspeicher und einer insbesondere peripheren Speichereinrichtung, in dem zum Indizieren und Speichern eines in einem mehrdimensionalen Würfel liegenden Datenbestands auf Speicherseiten gegebener Speicherkapazität des insbesondere peripheren Speichermittels ein wiederholtes iteratives Unterteilen des mehrdimensionalen Würfels in allen Dimensionen in Subwürfel erfolgt bis aufeinanderfolgende Subwürfel zu Regionen zusammenfaßbar sind die jeweils eine Menge von Datenobjekten beinhalten, die auf einer der Speicherseiten gegebener Speicherkapazität abspeicherbar sind.

2. Datenbanksystem nach Anspruch 1, in dem das Abspeichern der Datenobjekte einer Region auf einer Speicherseite gegebener Speicherkapazität unter Zuordnung eines Zeigers zu der Speicherseite und einer die Regionengrenzen bezeichnenden Adresse erfolgt.

3. Datenbanksystem nach Anspruch 2, in dem die Speicherung des Zeigers und der Adresse in einem B-Baum, B*-Baum oder Präfix-B-Baum erfolgt.

4. Datenbanksystem nach Anspruch 3, in dem die Speicherung der Datenobjekte in den Blattseiten des B-Baums, B*-Baums oder Präfix-B-Baum erfolgt.

5. Datenbanksystem nach einem der Ansprüche 2 bis 4, wobei die die Regionengrenzen bezeichnende Adresse aus Angaben über den letzten der die Region bildenden Subwürfeln besteht.

6. Datenbanksystem nach Anspruch 5, in dem die Adresse Angaben über die Anzahl der auf jeder Unterteilungsstufe in der Region enthaltenen Subwürfel umfaßt.

7. Datenbanksystem nach einem der Ansprüche 5 oder 6, in dem die Adresse aus einer Abfolge von dem jeweils letzten auf der jeweiligen Unterteilungsstufe noch von der Region vollständig eingeschlossenen Subwürfel zugeordneten Nummern besteht.

8. Verfahren zum Verwalten eines in einem n-dimensionalen Würfel mit $n > 1$ liegenden Datenbestands, wobei zum Indizieren und Speichern des Datenbestands auf Speicherseiten gegebener Speicherkapazität einer insbesondere peripheren Speichereinrichtung die folgenden Schritte ausgeführt werden:

1. Bilden einer Ausgangsregion bestehend aus dem Gesamtwürfel,
2. Zählen der in der Ausgangsregion vorhandenen Datenobjekte,
3. Abspeichern der Datenobjekte der Ausgangsregion auf einer Speicherseite, falls die Anzahl der Datenobjekte der Ausgangsregion nicht größer ist als die der vorgegebenen Speicherkapazität einer Speicherseite entsprechende Zahl von Datenobjekten,

4. andernfalls Auswählen einer Spaltadresse für die Ausgangsregion derart, daß durch Spalten der Ausgangsregion entlang der Spaltadresse eine erste und eine zweite Teilregion erzeugt werden, in welchen jeweils etwa die Hälfte der Datenobjekte der Ausgangsregion enthalten sind,
5. Fortführen des Verfahrens ab Schritt 2, wobei die erste Teilregion die Rolle der Ausgangsregion übernimmt,
6. Durchführen des Verfahrens ab Schritt 2, wobei die zweite Teilregion die Rolle der Ausgangsregion übernimmt.

9. Verfahren nach Anspruch 8, wobei das Abspeichern gemäß Schritt 3 unter paralleler Abspeicherung einer der betreffenden Region zugeordneten Adresse sowie eines der Adresse zugeordneten und auf die Speicherseite, auf der die gemäß Schritt 3 abgespeicherten Datenobjekte enthalten sind, verweisenden Zeigers erfolgt.

10. Verfahren nach Anspruch 9, bei dem die Speicherung der Adresse und des Zeigers in einem B-Baum, B*-Baum oder Präfix-B-Baum erfolgt, wobei durch aufeinanderfolgende Adressen jeweils Regionen definiert werden, deren Datenobjekte jeweils auf einer Speicherseite gegebener Speicherkapazität abgespeichert sind.

11. Verfahren zum Einfügen von Datenobjekten in einen insbesondere nach einem Verfahren gemäß einem der Ansprüche 8 bis 10 in Regionen aufgeteilten und auf Speicherseiten gegebener Speicherkapazität einer insbesondere peripheren Speichereinrichtung abgespeicherten n-dimensionalen Datenbestand mit $n > 1$ mit den folgenden Schritten:

1. Bestimmen einer dem einzufügenden Datenobjekt zugehörenden Region des n-dimensionalen Datenbestands aus den Koordinaten des Datenobjekts im n-dimensionalen Raum,
2. Auffinden der Speicherseite, auf der diese Region abgespeichert ist,
3. Bestimmen der Anzahl der auf dieser Speicherseite gespeicherten Datenobjekte,
4. Speichern des Datenobjekts auf dieser Speicherseite falls die Anzahl der gespeicherten Datenobjekte kleiner ist als die der gegebenen Speicherkapazität der Speicherseite entsprechende Anzahl,
5. andernfalls Auswählen einer Spaltadresse für die auf dieser Speicherseite gespeicherte Region derart daß durch Spalten der Region entlang der Spaltadresse eine erste und eine zweite Teilregion erzeugt werden in welchen jeweils weniger als etwa die Hälfte der der gegebenen Speicherkapazität entsprechenden Anzahl an Datenobjekten enthalten ist,
6. Einfügen des Datenobjekts in diejenige Teilregion, in der die Koordinaten des Datenobjekts liegen,
7. Speichern der ersten und der zweiten Teilregion auf jeweils einer Speicherseite.

12. Verfahren nach Anspruch 11, bei dem das Auffinden der Speicherseite gemäß Schritt 2 mittels in einem B-Baum, B*-Baum oder Präfix-B-Baum gespeicherter, den Speicherseiten zugeordneter Adressen und Zeiger erfolgt.

13. Verfahren nach Anspruch 12, bei dem das Speichern der ersten und der zweiten Teilregion gemäß Schritt 7 unter Ersetzung des Zeigers und der Adresse der Region aus Schritt 2 durch jeweils der ersten und der zweiten Teilregion zugeordnete Adressen und Zeiger erfolgt.

14. Verfahren nach Anspruch 13, bei dem die Speicherung der Adresse und des Zeigers in einem B-Baum, B*-Baum oder Präfix-B-Baum erfolgt, wobei durch aufeinanderfolgende Adressen jeweils Regionen definiert werden, deren Datenobjekte jeweils auf einer Speicherseite gegebener Speicherkapazität abgespeichert sind.

15. Verfahren zum Löschen von Datenobjekten in einem insbesondere nach einem Verfahren gemäß einem der Ansprüche 8 bis 10 in Regionen aufgeteilten und auf Speicherseiten gegebener Speicherkapazität einer insbesondere peripheren Speichereinrichtung abgespeicherten n-dimensionalen Datenbestand mit den folgenden Schritten:

1. Bestimmen einer dem zu löschenden Datenobjekt zugehörigen Region des n-dimensionalen Datenbestands aus den Koordinaten des Datenobjekts im n-dimensionalen Raum,
2. Auffinden der Speicherseite, auf der diese Region abgespeichert ist,
3. Löschen des Datenobjekts auf dieser Speicherseite,
4. Bestimmen der Anzahl der auf dieser Speicherseite gespeicherten Datenobjekte,
5. Verschmelzen der Region mit einer ihrer beiden Nachbarregionen falls die Anzahl der gespeicherten Datenobjekte kleiner ist als etwa die Hälfte der der gegebenen Speicherkapazität der Speicherseite entsprechenden Anzahl,
6. Bestimmen der Anzahl der in der durch das Verschmelzen neu entstandenen Region vorhandenen Datenobjekte,
7. Speichern der Region auf einer Speicherseite falls die Anzahl der in der Region vorhandenen Datenobjekte kleiner ist als die der gegebenen Speicherkapazität der Speicherseite entsprechende Anzahl,
8. andernfalls Auswählen einer Spaltadresse für die Region derart, daß durch Spalten entlang der Spaltadresse eine erste Teilregion und eine zweite Teilregion erzeugt werden, die jeweils etwa die Hälfte der gemäß Schritt 6 bestimmten Datenobjekte enthalten und Speichern der Teilregionen auf jeweils einer Speicherseite.

16. Verfahren nach Anspruch 15, wobei das Auffinden der Speicherseite gemäß Schritt 2 mittels in einem B-Baum, B*-Baum oder Präfix-B-Baum gespeicherter, den Speicherseiten zugeordneter Adressen und Zeiger erfolgt.

17. Verfahren zum Durchführen einer Datenanfrage auf der Grundlage eines gegebenen n-dimensionalen Anfragebereichs in einem insbesondere nach einem Verfahren gemäß einem der Ansprüche 8 bis 10 in aus Subwürfeln zusammengesetzte Regionen aufgeteilten und auf Speicherseiten gegebener Speicherkapazität

einer insbesondere peripheren Speichereinrichtung abgespeicherten n-dimensionalen Datenbestand mit den folgenden Schritten:

1. Bestimmen der Koordinaten des niedrigsten und des höchsten Schnittpunktes des Anfragebereichs mit dem n-dimensionalen Datenbestand,
2. Bestimmen derjenigen Region, in welcher der niedrigste Schnittpunkt liegt und Auffinden der Speicherseite, auf der diese Region abgespeichert ist,
3. Ermitteln sämtlicher auf dieser Speicherseite abgespeicherter Datenobjekte, die eine Schnittmenge mit dem Anfragebereich bilden und Ausgabe der ermittelten Datenobjekte,
4. Bestimmen des in der Abfolge letzten Subwürfels der in Schritt 2 bestimmten Region, der den Anfragebereich schneidet,
5. Beenden der Datenanfrage falls der höchste Schnittpunkt in diesem Subwürfel liegt,
6. andernfalls Ermitteln des nächsten Subwürfels derselben Ebene und desselben übergeordneten Würfels, der den Anfragebereich schneidet,
7. Bestimmen der Koordinaten des niedrigsten Schnittpunktes des Anfragebereichs mit dem ermittelten Subwürfel und Weiterführen des Verfahrens bei Schritt 2, falls in Schritt 6 ein Subwürfel ermittelt wurde,
8. andernfalls Ermitteln des nächsten Subwürfels der Ebene des übergeordneten Würfels aus Schritt 6, der den Anfragebereich schneidet und Durchführen von Schritt 6 mit den Subwürfeln des ermittelten Würfels,
9. Fortsetzen des Verfahrens mit Schritt 6, falls in Schritt 8 kein Subwürfel der Ebene des übergeordneten Würfels aus Schritt 6 ermittelt wurde, wobei der übergeordnete Würfel die Rolle des Subwürfels übernimmt.

Hierzu 6 Seite(n) Zeichnungen

– Leerseite –

Nummer: DE 196 35 429 A1
Int. Cl.⁶: G 06 F 17/30
Offenlegungstag: 5. März 1998

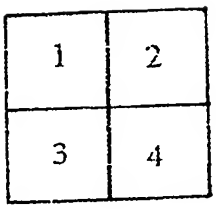


FIG. 1

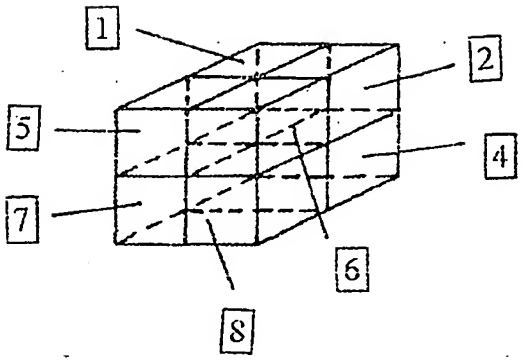


FIG. 2

Nummer: DE 196 35 429 A1
Int. Cl.[6]: G 06 F 17/30
Offenlegungstag: 5. März 1998

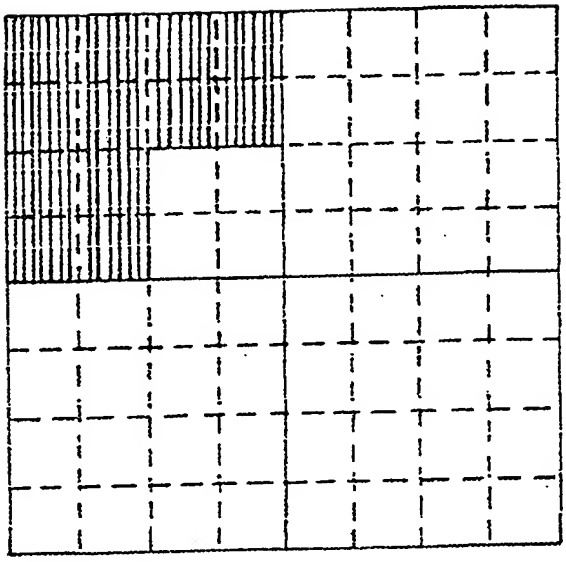


FIG. 3.1

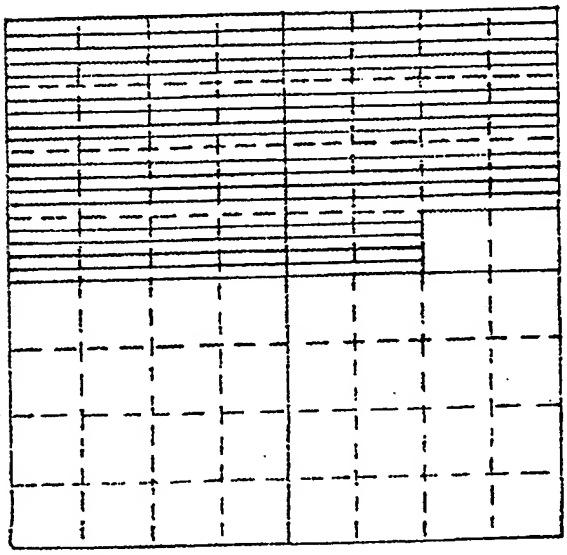


FIG. 3.2

Nummer: DE 196 35 429 A1
Int. Cl.[6]: G 06 F 17/30
Offenlegungstag: 5. März 1998

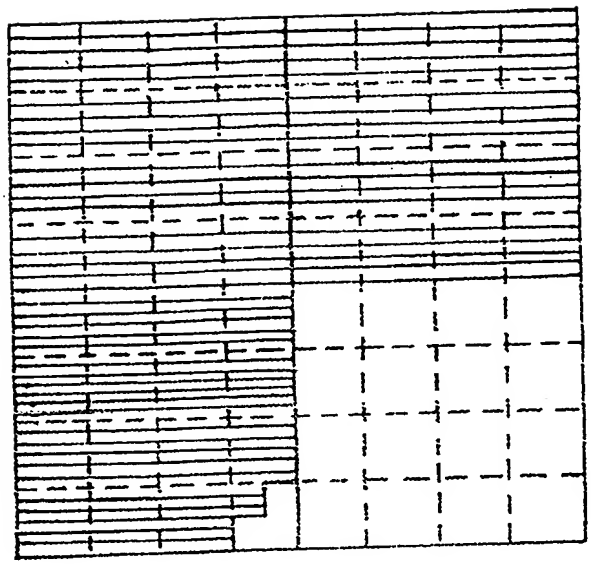


FIG. 3.3

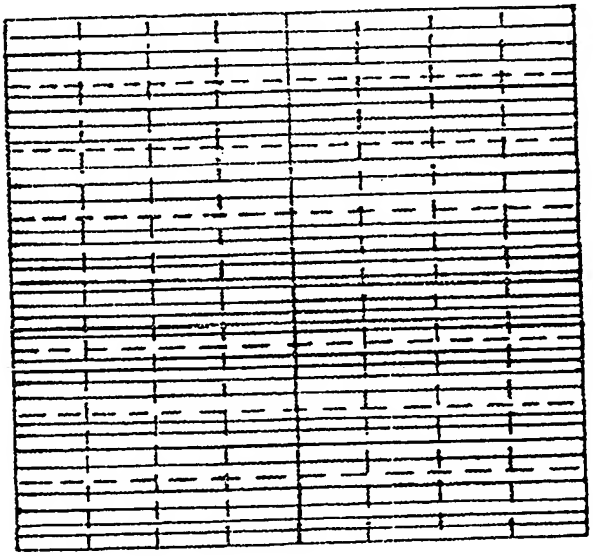


FIG. 3.4

Nummer: DE 196 35 429 A1
Int. Cl.[6]: G 06 F 17/30
Offenlegungstag: 5. März 1996

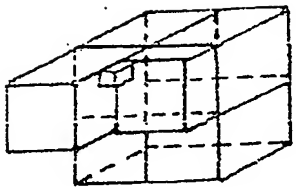


FIG. 4

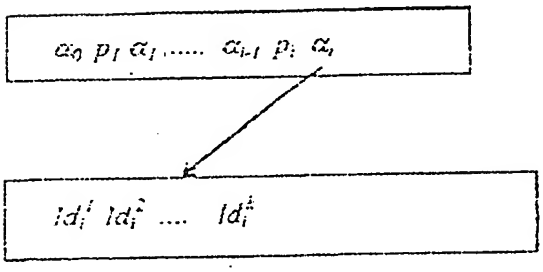


FIG. 5

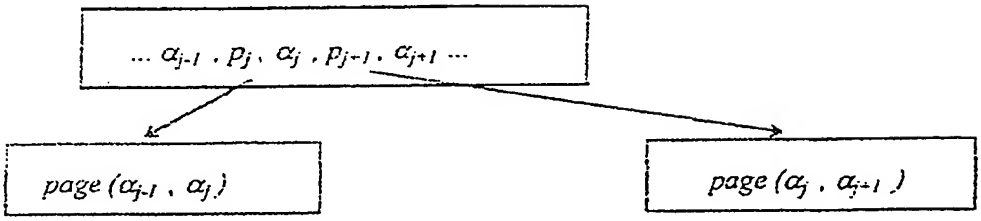


FIG. 6.1

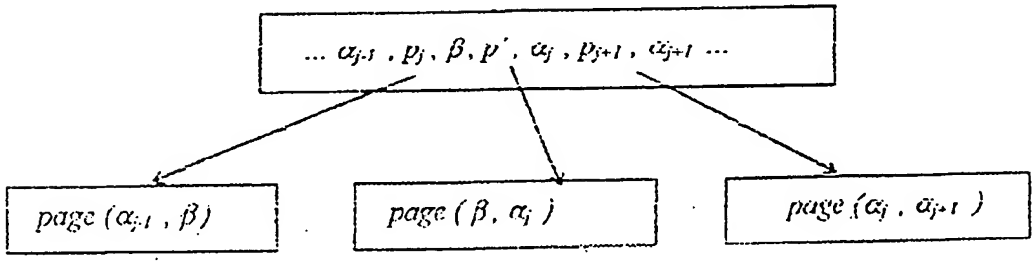


FIG. 6.2

Nummer: DE 196 35 429 A1
Int. Cl.[6]: G 06 F 17/30
Offenlegungstag: 5. März 1998

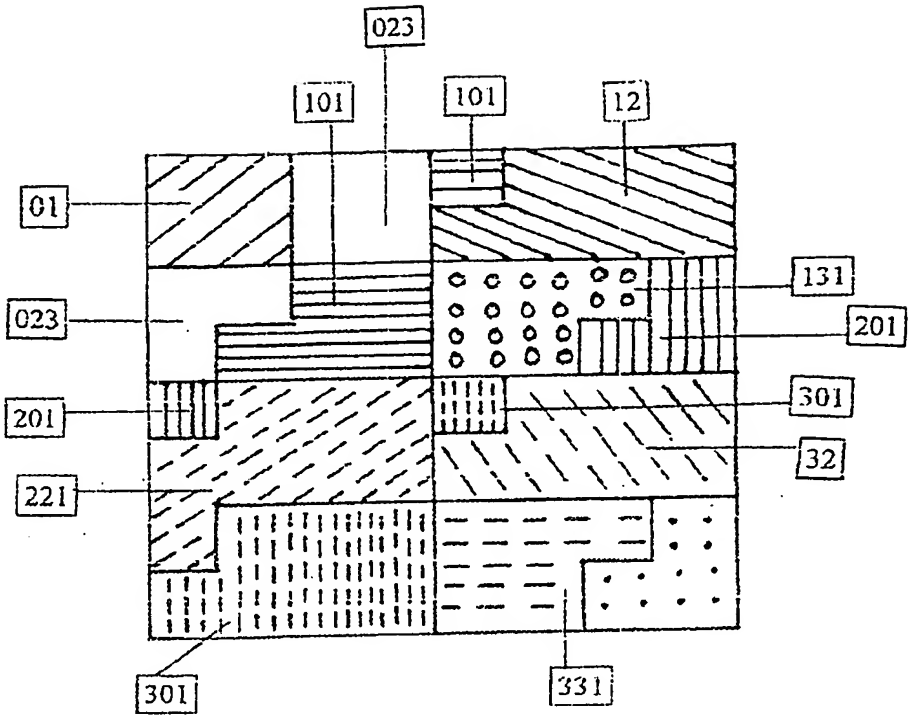


FIG. 7

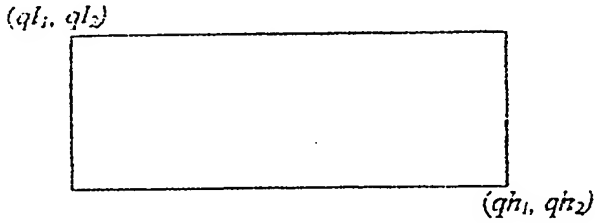


FIG. 8

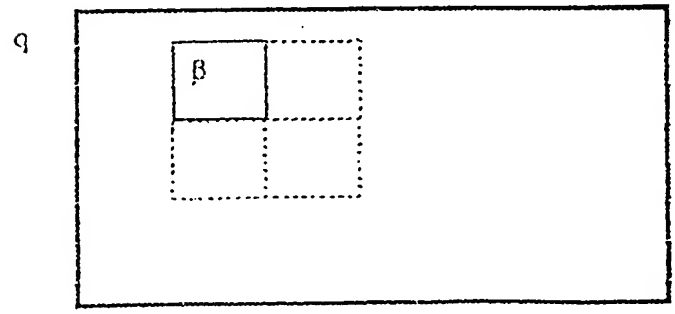


FIG. 9

Nummer: DE 196 35 429 A1
Int. Cl.[6]: G 06 F 17/30
Offenlegungstag: 5. März 1998

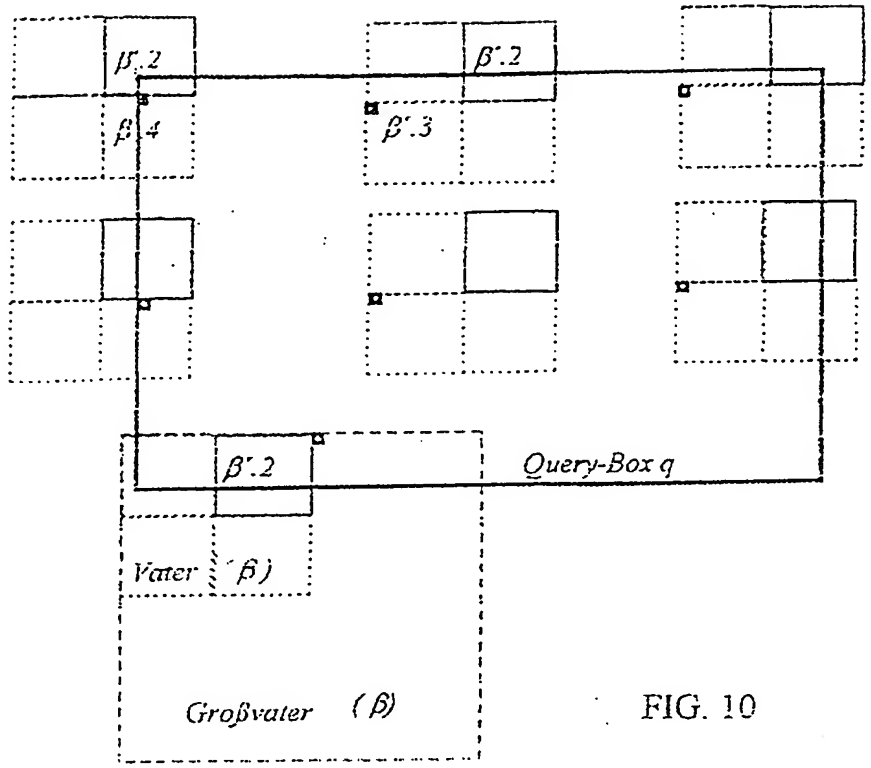


FIG. 10

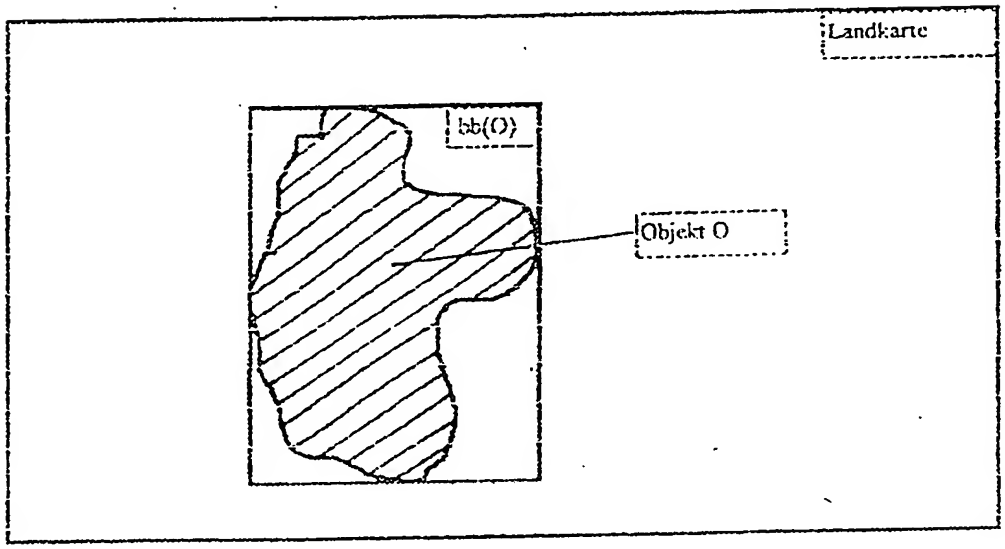


FIG. 11